# Linac Request Overload

*Analysis of problem*

Fri, Sep 11, 1998

During regular tuning of Linac magnet currents to optimize beam current and minimize beam losses, Larry Allen has often detected missing data points on his graphic displays on the "Linac console" in the Main Control Room. This console is served by a faster-than-typical Vax that was only 10% busy at the time of examination of the problem; hence, the Vax does not appear to be overloaded.

Two key graphic displays exhibited symptoms of missing data points. One was a display of 50 beam toroid and loss monitor readings, plotted at 15 Hz. With Linac beam cycles scheduled at 2–3 Hz, most of the data points are plotted near the base line, but it was often the case that beam cycles occurred without associated data points plotted. The second graphics display was a Fast Time Plot of 4 such signals plotted against the magnet power supply being tuned. Often some of the data points of the beam signals were missing.

The Vax supports a data pool that is managed by DPM, the Data Pool Manager. When new data arrives from the "source nodes," it is cached in this data pool. The application programs poll to collect the results. But the polling rate is not synchronized to 15 Hz. It may even be 60 or 70 ms, rather than 66 ms, as it is scheduled internally via VMS calls. If it should sometimes be more than 66 ms, it is possible that some data may be missed, even if it is being faithfully delivered at 15 Hz from the front ends. There is no attempt made to synchronize Vax application execution at the 15 Hz that the Linac uses. Whether this is what is primarily responsible for the observed missing data points is not so clear.

All Linac data is obtained via one of only two data server nodes, node0601 and node062E, which forward any request messages they received to a multicast address that reaches all Linac nodes. The real source nodes of data requests must deliver to the server nodes their contributions to any active requests by a deadline time, which is currently fixed at 40 ms beyond the start of the current 15 Hz cycle, as seen by the individual node, each of which is triggered to start cycle execution at 2-3 ms after Linac beam time. Some Linac nodes are busy updating their data pools and delivering their replies to any and all requesting nodes, so that they have trouble getting their replies sent to the server nodes much before the deadline time.

If a reply fails to reach a server node by the deadline time, the data that is delivered to a console will not include that data, of course. It might be expected to be included on the next cycle, or on the next time that the request is due for a reply, but because the timing is tight, the next reply data reaching the server might arrive before the deadline, which means it will overwrite the previous late data, so that the late data will therefore never make it to the Vax console.

Examining the network frame activity can show when frames are received or transmitted. During this period of heavy loading that was examined, when node062E was processing 370 messages per second, the network diagnostic showed that the replies from other Linac nodes arrived at node062E nearly always by the deadline time,

or shortly thereafter. But the servers have recently been configured to use IP when they forward requests via multicast to the other Linac nodes, so that the replies will necessarily be on IP as well. The recorded times associated with received network frames are optimistically early, because the diagnostic entry is recorded by the SNAP task, which supports the IP protocols. But the AcRq task, which processes RETDAT messages must execute in order for the reply data to be absorbed into the internal buffers used by each active request.

For the case examined, the times associated with sending the replies to consoles were in the 43–47 ms range, where under light loading conditions, they could be expected to be at 40–41 ms. Part of the delay in getting started delivering replies to consoles is due to processing late-arriving replies from the other nodes, but this also means that the required AcRq processing is even more delayed, perhaps even after the delivery of the replies to the consoles. It is necessary that reply data to servers be completely processedb by AcRq before the deadline time.

Actually, the situation is even worse than described. Before the AcRq task can execute, the ANet task, which interprets the Acnet header, must execute before the AcRq task can even be triggered.

Once the 40 ms deadline time interrupt occurs, an event is sent to the Server task to place it onto the ready queue. If the system is busy with SNAP processing at this time, for example, any subsequent Acnet header frames queued will cause ANet to run, but only after Server runs, which is too late.

Again, the only easy path to a solution is to move the deadline time to be somewhat later in the 15 Hz cycle. It must be late enough so that all related task activity with received reply messages sent to the server node can be completed. At the present time, this must be done as a patch in the code, as it is not designed to be modified easily. But the system code should be modified to make it easier to adjust the value of this deadline time.